Imperial College of Science, Technology and Medicine

Department of Computing

MSc C++ Programming – Assessed Exercise No. 2

Issued: Tuesday 2 November 2022
Due: **Before 19h00, Monday 21 November 2022**

The Exercise

Consider the configuration and behaviour of the traffic lights in Sleepy Town, as described below, and then

1. Draw the corresponding UML class diagram.
2. Write appropriate class definitions in C++ which, together with the main program given in the file `main.cpp`, generate a simulation of the specific scenario also described below.

General Information about Traffic Lights in Sleepy Town

1. Traffic lights exist in pairs of collaborating lights (one controlling the north-south traffic, the other one controlling the west-east traffic). Every traffic light has a colour (either red or green or yellow, and initially red), and a delay-time. Traffic lights turn from one colour to another. The delay time describes the time which a traffic light has to wait before it turns from one colour to an other. A traffic light is notified when a car wants to cross in the direction controlled by this light.
2. When a traffic light receives the message that a car wants to pass the crossing it is controlling, then
   (a) if it has red colour, and the collaborating light has green colour, it will request the collaborating light to turn to red.
   (b) if it has red colour, and the collaborating light has red colour, it will wait for its delay time, then turn to yellow, then again it will wait for its delay time, and then turn to green.
3. When a traffic light is requested to turn to red, then
   (a) if it has green colour it will wait for its delay time and then turn to yellow and request the collaborating light to turn to green.
   (b) if it has yellow colour, then it will wait for its delay time and then turn to red and request the collaborating light to turn to green.
4. When a traffic light is requested to turn to green, then
   (a) if it has red colour it will wait for its delay time and then turn to yellow and request the collaborating light to turn to red.
   (b) if it has yellow colour, then it will wait for its delay time and then turn to green.

5. A global clock keeps track of the time, and is augmented every time one of the traffic lights delays. Whenever a traffic light changes its colour, it displays an appropriate message with the time.

6. Time is measured in terms of hours (values from 0 to 23), minutes (values from 0 to 59), and seconds (values from 0 to 59). Remember, that adding a time given by `h1:m1:s1` to another time `h2:m2:s2` will give `h3:m3:s3`, where

   ```
   s3=(s1+s2)%60,
   m3=(m1+m2+((s1+s2)/60))%60
   h3=(h1+h2+(m1+m2+(s1+s2)/60)/60)%24
   ```

The Specific Scenario to Simulate

You have been given a main program which describes the following events (a) – (m):

a. First, the global clock is initialised to `0:0:0`, and

b. a pair of (slow) collaborating traffic lights T1 and T2 is set up, where T1 has a delay time of `0:15:0`, and T2 has a delay time of `0:5:0`.

c. Immediately, a car signals that it wants to cross in the direction controlled by T1.

d. Next, a car signals that it wants to cross in the direction controlled by T2.

e. Next, a car signals that it wants to cross in the direction controlled by T1.

f. Next, a car signals that it wants to cross in the direction controlled by T2.

g. Next, a pair of (ridiculously slow) collaborating traffic lights T3 and T4 is set up, where T3 has a delay time of `6:15:44`, and T4 has a delay time of `14:5:57`.

h. Immediately after this, a car signals that it wants to cross in the direction controlled by T3.

i. Next, a car signals that it wants to cross in the direction controlled by T3.

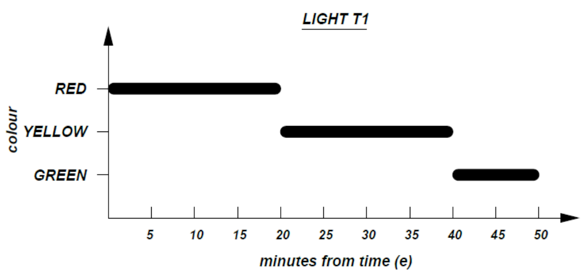
j. Next, a car signals that it wants to cross in the direction controlled by T4.

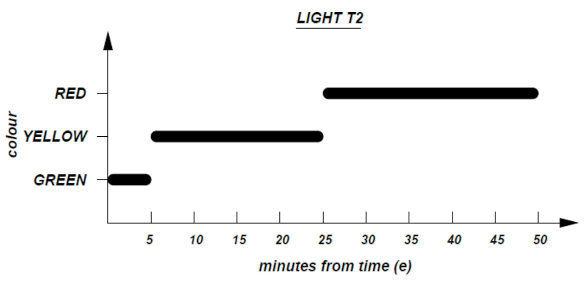
k. Next, a car signals that it wants to cross in the direction controlled by T4.

l. Next, a car signals that it wants to cross in the direction controlled by T3.

m. Finally, a car signals that it wants to cross in the direction controlled by T4.

Hence, for example, at point (e), when T1 receives the message that a car wants to cross, the following behaviour should be observable in the program output:





The Output

The output from the completed program should be similar to the following:

```
==========================
Roads open in Sleepy Town:
==========================

A pair T1 and T2 of (slow) collaborating lights is set up:

*** at 0:0:0 a car wants to cross light T1 (North South), with colour: red
    at 0:15:0 T1 (North South) changes colour to yellow
    at 0:30:0 T1 (North South) changes colour to green

*** at 0:30:0 a car wants to cross light T2 (East West), with colour: red
    at 0:45:0 T1 (North South) changes colour to yellow
    at 0:50:0 T2 (East West) changes colour to yellow
    at 1:5:0 T1 (North South) changes colour to red
    at 1:10:0 T2 (East West) changes colour to green

*** at 1:10:0 a car wants to cross light T1 (North South), with colour: red
    at 1:15:0 T2 (East West) changes colour to yellow
    at 1:30:0 T1 (North South) changes colour to yellow
    at 1:35:0 T2 (East West) changes colour to red
    at 1:50:0 T1 (North South) changes colour to green

*** at 1:50:0 a car wants to cross light T2 (East West), with colour: red
    at 2:5:0 T1 (North South) changes colour to yellow
    at 2:10:0 T2 (East West) changes colour to yellow
    at 2:25:0 T1 (North South) changes colour to red
    at 2:30:0 T2 (East West) changes colour to green

A new pair T3 and T4 of (very slow!) collaborating lights is now set up:

*** at 2:30:0 a car wants to cross light T3 (North South), with colour: red
    at 8:45:44 T3 (North South) changes colour to yellow
    at 15:1:28 T3 (North South) changes colour to green

*** at 15:1:28 a car wants to cross light T3 (North South), with colour: green

*** at 15:1:28 a car wants to cross light T4 (East West), with colour: red
    at 21:17:12 T3 (North South) changes colour to yellow
    at 11:23:9 T4 (East West) changes colour to yellow
    at 17:38:53 T3 (North South) changes colour to red
    at 7:44:50 T4 (East West) changes colour to green

*** at 7:44:50 a car wants to cross light T4 (East West), with colour: green

*** at 7:44:50 a car wants to cross light T3 (North South), with colour: red
    at 21:50:47 T4 (East West) changes colour to yellow
    at 4:6:31 T3 (North South) changes colour to yellow
    at 18:12:28 T4 (East West) changes colour to red
    at 0:28:12 T3 (North South) changes colour to green
```

```
*** at 0:28:12 a car wants to cross light T4 (East West), with colour: red
    at 6:43:56 T3 (North South) changes colour to yellow
    at 20:49:53 T4 (East West) changes colour to yellow
    at 3:5:37 T3 (North South) changes colour to red
    at 17:11:34 T4 (East West) changes colour to green

===================================
Roads close forever in Sleepy Town.
===================================
```

Approach

We suggest that you approach this exercise as follows:

1. Draw the UML diagram.
2. Complete the `Time` class by implementing the class definition in `time.cpp`, and test it.
3. Design the `TrafficLight` class, based on the UML diagram from step 1.
4. Complete the class declaration for the `TrafficLight` class in `trafficLight.h` and implement the class definition in `trafficLight.cpp`.
5. Test your solution using the main program in `main.cpp`.

For all the classes you define, you will have to ensure that your code includes appropriate declarations and definitions for the constructors and member functions used in calls within the main program `main.cpp`.

Submission

Place your function implementations in the files `time.cpp` and `trafficLight.cpp`, and any corresponding function declarations in the file `trafficLight.h`. Use the file `main.cpp` to test your functions. Create a `makefile` which compiles your submission into an executable file called `traffic`. Submit your final commit on LabTS. Your program files will be assessed partly by running them in an automated marking environment, and partly on their style, elegance, modularity, etc.

Acknowledgements

We acknowledge the contributions of Sophia Drossopoulou and Rob Miller to earlier versions of this exercise.